GPT-Fabric: Smoothing and Folding Fabric by Leveraging Pre-Trained Foundation Models

Vedant Raval*, Enyu Zhao*, Hejia Zhang, Stefanos Nikolaidis, and Daniel Seita

University of Southern California, Los Angeles, CA 90089, USA,
{ravalv, enyuzhao, seita}@usc.edu (* equal contribution)

Abstract. Fabric manipulation has applications in folding blankets, handling patient clothing, and protecting items with covers. It is challenging for robots to perform fabric manipulation since fabrics have infinite-dimensional configuration spaces, complex dynamics, and may be in folded or crumpled configurations with severe self-occlusions. Prior work on robotic fabric manipulation relies either on heavily engineered setups or learning-based approaches that create and train on robot-fabric interaction data. In this paper, we propose GPT-Fabric for the canonical tasks of fabric smoothing and folding, where GPT directly outputs an action informing a robot where to grasp and pull a fabric. We perform extensive experiments in simulation to test GPT-Fabric against prior methods for smoothing and folding. GPT-Fabric matches the state-of-the-art in fabric smoothing, and also achieves comparable performance with most prior fabric folding methods tested, even without explicitly training on a fabric-specific dataset (i.e., zero-shot manipulation). Furthermore, we apply GPT-Fabric in physical experiments over 10 smoothing and 12 folding rollouts. Our results suggest that GPT-Fabric is a promising approach for high-precision fabric manipulation tasks. Code, prompts, videos, and supplementary material are available at https://tinyurl.com/gptfab.

Keywords: Deformable Object Manipulation, Vision Language Models

1 Introduction

Robot fabric manipulation has potential to address a wide variety of real-world applications, including dressing assistance [8, 9], folding and unfolding laundry [6, 27], and manufacturing textiles [38]. However, robot fabric manipulation is challenging due to the infinite-dimensional configuration space of fabrics and the complex dynamics resulting from robot-fabric interaction [3, 34], which hinder the use of traditional motion planning techniques. Thus, fabric manipulation remains an active area of research, of which recent works have used machine learning to train fabric smoothing [35, 43] and folding [21, 41], with the intent of generalizing to different fabric configurations or targets. While these works show promising results, they require robot-fabric interaction data, either in simulation or in real. This data may consist of demonstrations [31], or "random" interaction data [23]. This raises the question of whether it is possible to get similar performance without creating or training on a fabric-related dataset.

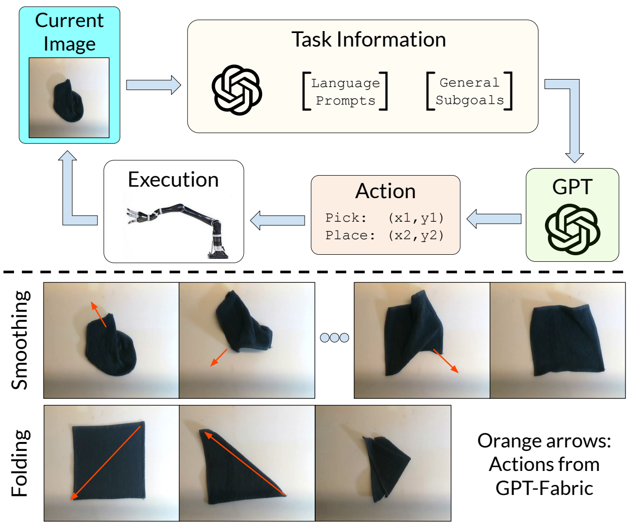


Fig. 1: **Top**: high-level overview of GPT-Fabric. The input to the foundation model (GPT) is the current image observation of the fabric and the task information. The latter might include fabric manipulation strategies generated by a VLM, natural language descriptions to prompt the foundation models, and (for folding tasks) the subgoal sequence targets (see Figure 2). GPT-Fabric directly produces actions (e.g., pick-and-place) for a robot. **Bottom**: example rollouts of GPT-Fabric for smoothing and folding.

In parallel to these works on fabric manipulation, the AI research community has recently experienced an explosion of interest in foundation models [2] such as GPT-4 [32] and Gemini [12]. These models are trained on broad data and are capable of "zero-shot" generation of images, language, and code. In robotics, a standard way to employ foundation models is as a high-level planner. For example, [37] and [22] use large language models to generate code describing high-level steps for a robot to execute. Consider when a robot must put food in a refrigerator; these approaches may produce code specifying: (1) go to kitchen, (2) open refrigerator, (3) go to counter, (4) get food, and (5) put in refrigerator, but this abstracts away low-level control. How, then, can we leverage the broad knowledge inherent in such foundation models for complex, low-level deformable object manipulation?

In this paper, we present GPT-Fabric (Figure 1), that uses OpenAI's GPT to directly output low-level manipulation actions for robotic fabric manipulation. Despite the impressive performance of GPT on various tasks [32], it is nontrivial to apply GPT to complex fabric manipulation tasks. As we later show in ablation experiments, we obtain poor fabric manipulation performance by naively providing the fabric image and prompting GPT. With GPT-Fabric, we show how to more effectively use GPT for fabric manipulation, and our method obtains comparable performance to most prior works and does not require creating a fabric-related dataset. More broadly, we hope this research helps facilitate using pre-trained foundation models for high-precision deformable object manipulation tasks.

In summary, this paper contributes:

1. GPT-Fabric, a novel method for robot fabric manipulation that leverages GPT for low-level decision-making.
2. State-of-the-art fabric smoothing and comparable fabric folding performance to most prior works, without needing to create a training dataset.
3. Ablation experiments investigating the importance of different aspects of GPT-Fabric.

2 Related Work

2.1 Fabric Manipulation

Fabric smoothing and folding are two canonical robotic fabric manipulation tasks. Pioneering research in this area has used bimanual robots with geometric algorithms. For example, a common approach to smooth fabrics was to leverage gravity to naturally smooth out the fabric to reveal corners, resulting in high success rates in smoothing crumpled towels [6, 27]. Other researchers have studied how to grasp or smooth fabrics by utilizing geometric features, such as corner detection [42] or fitting contours to clothing [30]. While effective, these prior approaches may require strict hardware, be time-consuming, or may have limited generalization to diverse fabric configurations.

Deep learning in robotics has renewed interest in fabric manipulation, with the intent of leveraging powerful function approximators to learn complex motor skills from high dimensional observational data. Researchers have demonstrated fabric smoothing and folding using either imitation learning [15, 35] or reinforcement learning [14, 43], or both [33]. Other complementary techniques for data-driven fabric manipulation include latent space planning [25, 45], learning dense visual descriptors [11] and leveraging dynamic manipulation [4, 44].

Methods that use learning, however, require suitable fabric interaction data, which can be obtained via demonstrations [31], simulators [1, 24], or from real world trials [21]. A naturally related question is whether one can reduce the data requirement for learning fabric manipulation. Prior work has explored this by leveraging appropriate *representations.* For example, [23] show that learning with a particle-based fabric representation is more sample-efficient than learning from images or a latent space for smoothing, and [41] show a similar benefit for fabric folding by leveraging optical flow. Eliminating the need for goal observations specific to each fabric configuration, [31] improve fabric folding results by employing space-time attention in a Transformer architecture. Nonetheless, these prior works still require creating and training on fabric-related interaction data. In contrast, we propose a novel approach to leverage the broad knowledge in GPT to attain competitive fabric manipulation performance which avoids the need to create fabric-related training data. Our method also uses natural language to prompt GPT to directly output low-level actions, distinguishing it from work that uses pre-trained language embeddings to describe folding targets [5].

2.2 Foundation Models in Robotics

The emergence of foundation models [2] such as GPT-4 [32] and Gemini [12], has revolutionized the field of Artificial Intelligence. These models, which include Large Language Models (LLMs), are trained on massive data at scale, and can be deployed for many downstream applications. While these foundation models are often used for text-related applications such as natural language generation, researchers have recently employed them for real-world *robotics* tasks; see [10, 16, 19] for representative surveys. One way to use foundation models is as a high-level planner, which can generate a sequence of steps that a robot should take to achieve an objective [7, 17]. Building on this, prior work has also leveraged foundation models to generate code to specify a robot's policy [18, 22, 37]. In addition, these models can generate reward functions for robots [26, 47], potentially by querying pairs of images of an agent's observations [40].

Along with using foundation models for high-level tasks, there has been some recent work using them to specify low-level actions for decision-making. For example, [39] demonstrate how to prompt GPT-4 to produce joint angles to make a quadruped robot walk. [28] and [29] use GPT models by recasting autonomous driving as a language modeling problem. Finally, among the most relevant recent works, [20] use GPT as a zero-shot planner to generate a dense sequence of end-effector robot poses, and benchmark this on a variety of language-conditioned tasks. Taking inspiration from these, we study the novel application of applying foundation models for complex, low-level deformable object manipulation.

3 Problem Statement

In this paper, we study quasi-static fabric smoothing and folding with a single-arm robot. We assume that there is one piece of fabric on a flat workspace. We use $\boldsymbol{\xi}_t$ and $\mathbf{o}_t$ to represent, respectively, the fabric *state* and *image observation* at time t in an interaction *rollout*, which lasts up to a user-specified value of at-most T time steps. A rollout transforms the initial fabric state $\boldsymbol{\xi}_0$ and image $\mathbf{o}_0$ to the final state $\boldsymbol{\xi}_T$ and image $\mathbf{o}_T$, respectively. More concretely, $\boldsymbol{\xi}_t \in \mathbb{R}^{N \times 3}$ is the set of N particles (or "points") that form the fabric. Each particle has a 3D world coordinate position. In addition, $\mathbf{o}_t \in \mathbb{R}^{H \times W \times c}$ is an RGB-D image with height H, width W, and $c = 4$ channels (three for color, one for depth).

We specify a robot's actions with picking and placing poses: $\mathbf{a}_t = \{x_{\text{pick}}, x_{\text{place}}\}$, where the robot grasps the fabric at x_{pick}, lifts the fabric by a small amount, drags it parallel to the workspace plane towards x_{place}, where it then releases the fabric. Here, x_{pick} and x_{place} are 3D world coordinate positions. They may be derived from 2D image pixels p_{pick} and p_{place} via robot-camera calibration. This type of action primitive is standard in prior work on (quasi-static) robot fabric manipulation [14, 23, 35, 43].

Finally, we assume access to a corner detector which is *not* fabric-specific (e.g., Harris [13] or Shi-Tomasi [36]).

Given this formulation, we consider the following tasks.

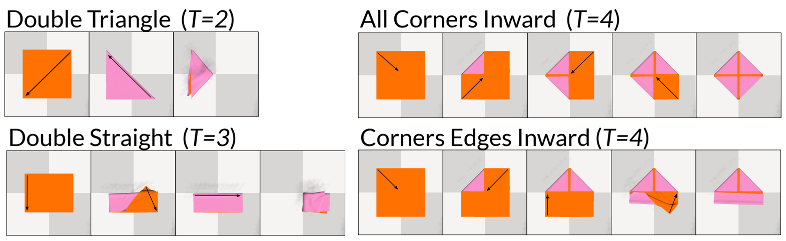


Fig. 2: Subgoal sequences for the four folding tasks we consider (from [31]) with maximum rollout lengths T. We use these for folding in simulation (Section 5.2) and real (Section 7).

Smoothing The fabric starts in a crumpled configuration. We assume a priori that the objective is smoothing. As in prior work on smoothing [23, 35], the objective is to improve *fabric coverage* $C(\boldsymbol{\xi}_t) \in [0,1]$, which is the fraction of the fabric's area when projected on the 2D workspace xy-plane, compared to the area of a fully flat fabric. We specify x_{pick} as one of the N points of the fabric state $\boldsymbol{\xi}_t$. We consider policies that specify the change in the x and y coordinates after x_{pick}, which enables us to obtain x_{place}. For experiments, we test maximum rollout lengths of $T \in \{5, 10, 20\}$ for consistency with [23].

Folding The fabric starts flat on the workspace. As in prior work [31, 41], we assume access to a sequence of T subgoal image observations $\{\mathbf{o}_1^{(g)}, \mathbf{o}_2^{(g)}, \ldots, \mathbf{o}_T^{(g)}\}$ representing the process to obtain the final desired fabric configuration for each folding task, since a single goal observation may not describe the full fabric state due to self occlusion. Going from each subgoal to the next is possible with a single pick-and-place action, however we do not assume access to this ground-truth action. Although a natural language description for each folding step could benefit a VLM-based method like ours, we do not assume access to this information to remain consistent with prior work.

Following [31], we use a generic demonstration subgoal sequence for each task used for different starting fabric configurations (see Figure 2). We add an arrow-based action visualization to aid our method in visual reasoning. Note that these arrows do not represent the ground-truth action at test time.

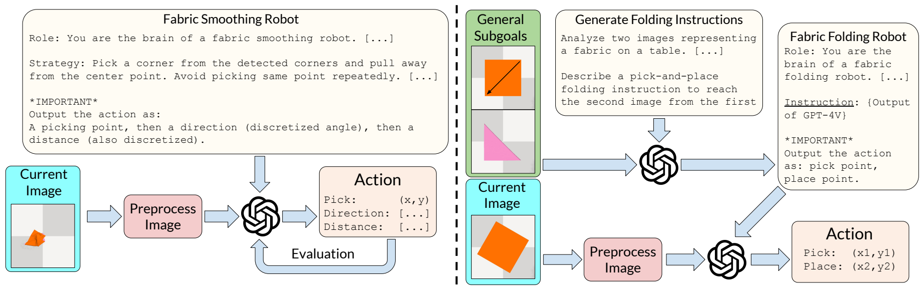
We assume that at test time, the robot must execute a novel sequence of subgoals, and we input $\mathbf{o}_t^{(g)}$ and $\mathbf{o}_{t+1}^{(g)}$ to the robot to reach the fabric state $\boldsymbol{\xi}_{t+1}$. We represent the pick p_{pick} and place p_{place} points as 2D image coordinates in the image $\mathbf{o}_t$ of the fabric.

In simulation, we evaluate using *mean particle distance error* between the achieved and goal fabrics [31, 41]. In real world, without ground-truth particle information, we evaluate by human inspection as done in prior work [11, 14, 21].

4 Method

4.1 GPT-Fabric: Overall Structure

As suggested in prior work [14, 23, 35, 41], the corners and center of a fabric are the most relevant for a successful fabric manipulation. Thus, for each time t, we



```
Role: You are the brain of a fabric smoothing robot. [...]

Strategy: Pick a corner from the detected corners and pull away
from the center point. Avoid picking same point repeatedly. [...]

*IMPORTANT*
Output the action as:
A picking point, then a direction (discretized angle), then a
distance (also discretized).
```

```
Pick:          (x,y)
Direction:  [...]
Distance:   [...]
```

```
Analyze two images representing
a fabric on a table. [...]

Describe a pick-and-place
folding instruction to reach
the second image from the first
```

```
Role: You are the
brain of a fabric
folding robot. [...]

Instruction: {Output
of GPT-4V}

*IMPORTANT*
Output the action
as: pick point,
place point.
```

```
Pick:   (x1,y1)
Place:  (x2,y2)
```

Fig. 3: Our GPT-Fabric method, for smoothing (left) and folding (right). The input prompt to GPT includes the current RGB-D image of the fabric $\mathbf{o}_t$ (only showing RGB above) and the task information. For folding, this information includes subgoal sequences (see Figure 2). We also preprocess $\mathbf{o}_t$ for GPT-Fabric and use an *evaluation module* for smoothing; see Figure 4 for details.

preprocess the image observation $\mathbf{o}_t$ to extract these keypoints to be part of the prompt to GPT. While the exact preprocessing of $\mathbf{o}_t$ and prompting are different for smoothing and folding (see Sections 4.2 and 4.3), both use depth to mask out the background, and off-the-shelf corner detectors (as per our assumptions in Section 3) to obtain estimates of the fabric corners. These detectors are not fully reliable and could cause errors. This processed information is part of the prompts. If the prompt does not include in-context examples, then we refer to this method as **GPT-Fabric (zero-shot)**.

Given the prompt, GPT produces a pick pose as a 2D pixel p_{pick} in the image representation $\mathbf{o}_t$, which we convert to x_{pick}. GPT also produces a place pose as a 2D pixel p_{place} for folding or as a tuple of the moving direction and the moving pixel distance for smoothing. In both cases, we convert this to x_{place}. The robot, either in simulation or in real, can utilize x_{pick} and x_{place} to perform the action.

4.2 GPT-Fabric for Smoothing

We use Shi-Tomasi Corner Detection [36] to detect possible fabric corners, and approximate the center of the fabric via its bounding box. Annotation can improve GPT's reasoning about images [46], so we annotate the RGB image with the detected corners, bounding box, and approximate fabric center. For all time steps after the initial step, we also annotate the RGB image with the prior action's *placing point* location. Figure 4 visualizes the above preprocessing and annotation procedure for smoothing. After obtaining the annotated image, we combine it with a natural language prompt for the foundation model GPT-4V, as visualized in Figure 3 (left). The language prompt describes the smoothing task, an explanation of the annotated RGB image, and a high-level strategy. The prompt decomposes the strategy into low-level instructions to guide GPT-4V to:

- pick one of the detected corners as the picking point.
- pick a pull orientation angle that moves "outwards" from the fabric's center.
- pick the largest pull distance which avoids dragging the fabric center too far from the image center.

We discretize pull orientations and distances to simplify the task and request a structured output for the action. The following is a summary of the prompts:

```
- Your task is to provide a pick and place action with the following information:
1. The pick location. Pick one of the given corners {...}. # GPT-4V's answer here
2. The pull angle. Pick one of {0,pi/4, pi/2,..., 7*pi/4}. # GPT-4V's answer here
3. The pull length. Pick one {0.1, 0.25, 0.5, 0.75, 1.0},where 1 is the length of
   a flattened fabric edge. # GPT-4V's answer here
```

Given the proposed action, we use an *evaluation module* to verify if the action follows our high-level strategy (see Figure 4). This module checks the picking point and its Euclidean distance to the prior picking point, as well as the orientation. We execute the action if it satisfies our tests, and otherwise query GPT again for another proposed action. We limit to three queries per time step. Instead of filtering detected corners by distance and orientation, the evaluation module guides GPT to learn from its mistakes and improve smoothing, avoiding over-engineering by not directly supplying the optimal corners.

4.3 GPT-Fabric for Folding

At each time t, we use GPT-4V to analyze the current subgoals $\mathbf{o}_t^{(g)}$ and $\mathbf{o}_{t+1}^{(g)}$ and provide a natural language folding instruction. We also preprocess the RGB-D image $\mathbf{o}_t$ to obtain a set of candidate corners via Harris Corner Detection [13] and use the fabric mask to estimate the fabric center. This generated instruction, along with the detected key-points, is passed to the foundation model (either GPT-3.5 or GPT-4) to return a pick-and-place policy to achieve the desired subgoal. We construct a system prompt which offers a universal context related to fabric folding, inspired by [28]. By doing this, we do *not* hand-design a lengthy prompt that is specific to a folding task. Instead, GPT-4V produces the instructions for each type of fold. If we want to achieve a new folding task, it should be possible to use GPT-Fabric simply by specifying the folding subgoal sequence. The following is a summary of the prompts:

```
- Given two folding subgoal images with an arrow representing a pick-and-place action {...},
provide an instruction to achieve the transition between the images. # GPT-4V's answer here
- Given the above folding instruction, provide a pair of pick and place points on the cloth
from the corners {...} to achieve this folding step. # GPT-4 or GPT-3.5's answer here
```

This system does not require expert demonstrations and can act as a zero-shot manipulator. However, we can leverage the in-context learning capabilities of GPT by providing demonstrations while generating folding instructions as well as while getting the pick and place points. Figure 3 (right) visualizes the folding pipeline. If we use in-context learning, we call this method **GPT-Fabric (in-context)**. Given the limitations of the fine-tuning API to support only text, we leave a *GPT-Fabric (fine-tuned)* extension to future work.

5 Simulation Experiments

For consistent comparisons with prior works, we leverage SoftGym simulation [24] to test GPT-Fabric.

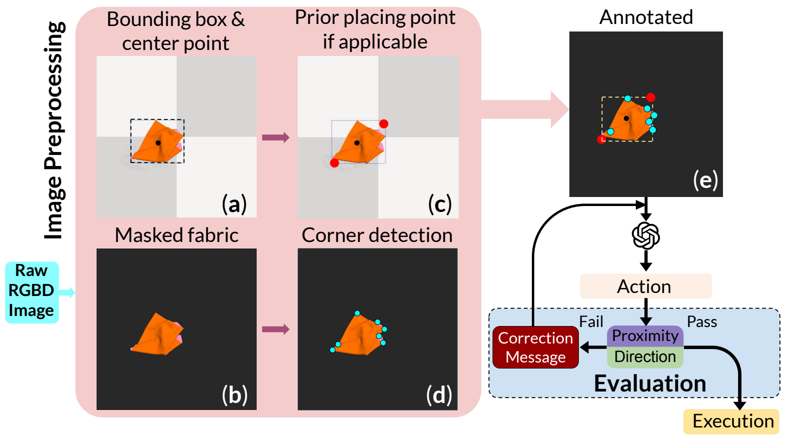


Fig. 4: Details of image annotation and evaluation for smoothing (see Section 4.2 for more). From RGB-D image $\mathbf{o}_t$, we get a bounding box and approximate fabric center (a) via masking (b). If applicable, we annotate the *prior* placing point and its "symmetric" point about the fabric center (c). We detect corners (d) on the masked image, then combine (c) and (d) to get image (e) as input to GPT-4V. We use an evaluation module to verify GPT's output. If it fails, we ask GPT to try again with a correction message.

5.1 Fabric Smoothing Experiments

We compare GPT-Fabric for single-arm, quasi-static fabric smoothing with data and results from Visible Connectivity Dynamics (VCD) [23]. Lin et al. [23] showed that VCD attains superior coverage performance compared to Fabric-VSF [14], Contrastive Forward Model [45], and Maximal Value under Placing [43]. Thus, we use the same square fabric in the 40 starting, crumpled configurations from [23] for testing GPT-Fabric on smoothing. To compare results, we directly use statistics reported in [23]. In addition, we also download and deploy their publicly released trained model weights on the same fabric configurations. We perform one rollout for zero-shot GPT-Fabric for each of the 40 starting fabric configurations and each of the three maximum possible allowed actions (5, 10, and 20), for a total of 120 rollouts.

We evaluate using the Normalized Improvement (NI) in coverage. NI computes the increased covered area normalized by the maximum possible improvement, $NI = \frac{s - s_0}{s_{max} - s_0}$, where s_0, s, and s_{max} are the initial, achieved, and maximum possible fabric coverage. As in [23], smoothing rollouts end early when NI exceeds 0.95. In our quantitative smoothing results, to be consistent with [23], we report the 25%, 50%, and 75% percentiles (Q_{25}, Q_{50}, Q_{75}) of NI; the number before $\pm$ is Q_{50} and the number after $\pm$ is $\max(|Q_{50} - Q_{25}|, |Q_{75} - Q_{50}|)$.

5.2 Fabric Folding Experiments

We primarily compare GPT-Fabric for single-arm, quasi-static fabric folding with data and results from Foldsformer [31]. Mo et al. [31] performed controlled experiments for fabric folding, and showed that Foldsformer achieves the lowest (i.e., best) mean particle distance error compared to FabricFlowNet [41], Fabric-VSF [14], and Lee et al. [21]. Thus, we leverage the open-source Foldsformer data for our experiments, and we directly use results that they report for comparisons. We test on their same starting fabric configurations, their same (test-time) subgoal sequences for folding, and the same folding types (see Figure 2). For *Double Triangle*, *All Corners Inward*, and *Corners Edges Inward*, we use a square fabric in 40 different configurations per task, combining 10 distinct side lengths, ranging from 31.25cm to 36.875cm, and 4 rotation angles of 0°, 30°, 45°, and 60°. For *Double Straight*, we use a rectangular fabric with the 10 sizes ranging from 31.25cm×27.5cm to 36.875cm×32.5cm and the above mentioned rotation angles.

We set the temperature of GPT-4V to 0.2 to allow some diversity in the generated instructions. We consider both GPT-4 and GPT-3.5 to reason from these instructions while performing experiments for zero-shot as well as in-context version. For in-context learning, we generate demonstrations (see Section 5.3) by leveraging a script provided by Foldsformer [31]. Since the GPT models are not deterministic, we run GPT-Fabric for five trials for each fabric starting state and subgoal sequence. We average results from those five to get metrics for a single starting state and target combination. We then average over all those combinations to obtain our final mean particle distance error results. To be consistent with [31], for our quantitative results, the number before ± is the mean and the number after is the standard deviation.

5.3 Data Sizes

One of our motivations is to leverage the broad knowledge in GPT to avoid explicitly creating a dataset for fabric manipulation. We measure a method's "data size" by the amount of pick-and-place actions in its training data, where the only requirement is that the actions are applied on fabric. We do not distinguish between an action specific to a fabric manipulation task (e.g., specific to smoothing, or to a *Double Triangle* fold) or a "random" pick-and-place action on a fabric. These count the same towards the overall data size.

For smoothing, VCD [23] trained on 2000 pick-and-place actions. This is orders of magnitude smaller than Fabric-VSF [14] trained on 106,725 actions, Contrastive Forward Module [45] trained on 400,000 actions, and Maximal Value of Placing [43], trained on 250,000 actions. Lin et al. [23] benchmarked these prior methods using data sizes similar to those reported in the original papers.

For folding, FabricFlowNet [41] trained on 20,000 actions in simulation, Fabric-VSF [14] used 106,725 actions in simulation, and Lee et al. [21] only used real world data consisting of 300 actions. Mo et al. [31] benchmarked these prior methods and their proposed Foldsformer method on a common dataset of 48,000 random actions in simulation. In addition, they used the same 100 task-specific demonstrations for all methods, adding more 200-400 actions per task.

In contrast, GPT-Fabric (zero-shot) does not require creating a fabric-related interaction data. Moreover, we only consider 10 expert demonstrations per each action in each folding task for the in-context version of GPT-Fabric, adding 20 actions for Double Triangle, 30 actions for Double Straight, and 40 actions each for All Corners Inward and Corners Edges Inward, totalling to only 130 actions.

Method	# of actions: 05	# of actions: 10	# of actions: 20	Data Size
VCD[†]	0.624 ± 0.217	0.778 ± 0.222	0.968 ± 0.307	2K
VCD[§]	0.472 ± 0.205	0.625 ± 0.222	0.791 ± 0.281	2K
VCD, graph imitation[†]	0.692 ± 0.258	0.919 ± 0.377	0.990 ± 0.122	2K
Fabric-VSF[†]	0.321 ± 0.112	0.561 ± 0.127	0.767 ± 0.134	∼105K
CFM[†]	0.053 ± 0.051	0.077 ± 0.053	0.109 ± 0.066	400K
MVP[†]	0.399 ± 0.210	0.435 ± 0.137	0.421 ± 0.361	250K
GPT-Fabric (zero-shot)	0.733 ± 0.229	0.959 ± 0.240	0.986 ± 0.035	0

[†]Reported by VCD authors. [§]Using trained model weights provided by VCD authors.

Table 1: Results for fabric smoothing in simulation, comparing GPT-Fabric versus prior works on the same starting crumpled (square) fabric configurations from [23]. We report $Q_{50} \pm \max(|Q_{50} - Q_{25}|, |Q_{75} - Q_{50}|)$ for the normalized improvement in coverage; see Section 5.1 for details. This is *not* the same statistic as we use in Table 2.

Method	Double Triangle	Double Straight	All Corners Inward	Corners Edges Inward	Data Size
Foldsformer[†]	19.64 ± 17.08	59.09 ± 44.82	3.06 ± 1.83	8.11 ± 7.96	>48K
Foldsformer[§]	14.00 ± 15.39	28.79 ± 30.65	2.94 ± 1.74	8.49 ± 2.20	>48K
FabricFlowNet[†]	102.20 ± 19.39	85.34 ± 19.67	33.64 ± 11.76	36.95 ± 9.95	>48K
Lee et al.[†]	109.82 ± 39.96	114.71 ± 26.06	27.21 ± 11.47	70.67 ± 22.24	>48K
Fabric-VSF[†]	114.62 ± 35.46	116.94 ± 24.52	46.05 ± 23.38	51.82 ± 16.65	>48K
GPT-Fabric (GPT-4, zs)	82.61 ± 6.74	76.63 ± 8.40	63.51 ± 12.84	79.94 ± 22.51	0
GPT-Fabric (GPT-3.5, zs)	84.76 ± 8.51	73.24 ± 7.51	67.70 ± 12.31	98.99 ± 25.28	0
GPT-Fabric (GPT-4, ic)	43.43 ± 17.44	72.56 ± 13.57	57.53 ± 20.27	121.98 ± 15.44	130
GPT-Fabric (GPT-3.5, ic)	51.89 ± 18.55	69.24 ± 9.68	61.20 ± 18.93	128.57 ± 22.25	130

[†]Reported by Foldsformer authors. [§]Using trained model weights provided by Foldsformer authors.

Table 2: Results for fabric folding in simulation, comparing GPT-Fabric versus prior works on the same folding subgoal targets from Mo et al. [31]. We report the mean particle distance error (mm) for the four fold types, and the number after ± is the standard deviation. For GPT-Fabric, "zs" is "zero-shot" and "ic" is "in-context."

6 Simulation Results

6.1 Fabric Smoothing Results

Figure 5 (left) shows qualitative results of zero-shot GPT-Fabric for smoothing a square fabric in simulation. Table 1 reports the normalized improvement (NI) in coverage, for different total allowed pick-and-place actions, using numbers directly from Supplementary Table 4 in [23] for all baselines. Due to space limitations, we defer additional metrics to the supplementary material.

The results show GPT-Fabric (zero-shot) matches or outperforms the best baseline (VCD with graph imitation), with a higher median for 5-action and 10-action horizons, while VCD has a higher median NI for 20-action horizon. Since GPT-Fabric (zero-shot) outperforms the baselines, we do not include an in-context version for smoothing.

Ablation Studies We ablate components of GPT-Fabric and report the NI in smoothing in Table 3, using the same notation as in Table 1. We test four

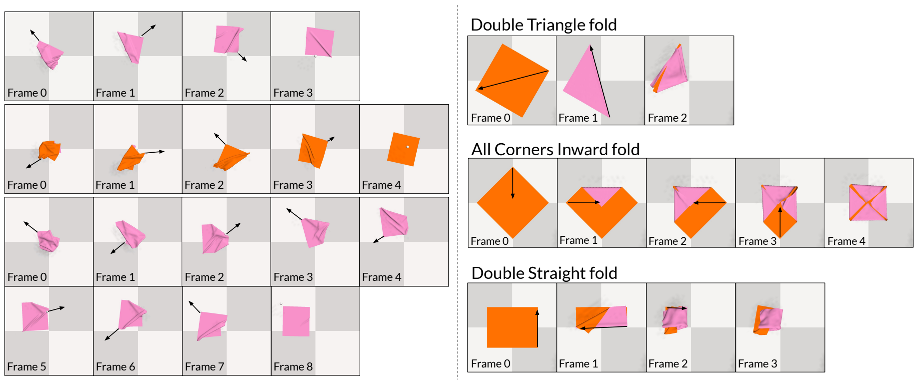


Fig. 5: Qualitative results for smoothing (left) and folding (right) from GPT-Fabric in SoftGym simulation. We show three smoothing rollouts of varying frame lengths, where each frame shows one pick-and-place action. In all three smoothing rollouts, GPT-Fabric achieved NI>0.95. We show three examples of folding rollouts with different folding subgoals (see Figure 2). GPT-Fabric was unable to achieve qualitatively good results for *Corners Edges Inward*.

Method	NI
Random discretized action	0.203 ± 0.230
No image preprocessing & no eval. module	0.021 ± 0.144
No eval. module	0.540 ± 0.220
No verifying pick point proximity in eval. module	0.682 ± 0.222
GPT-Fabric (zero-shot)	0.733 ± 0.229

Table 3: Normalized improvement (NI) for GPT-Fabric ablations for smoothing in simulation after 5 pick-and-place actions. The bottom row is the full GPT-Fabric method (reported in Table 1).

ablations: (1) random selection of action parameters, (2) no image preprocessing of $\mathbf{o}_t$, (3) no evaluation ("eval.") module to verify GPT's output, or (4) only using part of it by removing the proximity check (see Figure 4). The results indicate that removing components of GPT-Fabric leads to worse performance. See the supplementary material for more details.

6.2 Fabric Folding Results

Figure 5 (right) shows qualitative results of GPT-Fabric for fabric folding in simulation. We report results in Table 2, where the values from baseline methods are directly from Table I in Mo et al. [31]. Overall, the results suggest that GPT-Fabric attains competitive performance compared to prior works of FabricFlowNet and Fabric-VSF, even as a zero-shot manipulator. While results are worse than Foldsformer, we use a fraction of the data (in the in-context version) and we do not need to train a model on fabric manipulation data.

We observe that for *Double Triangle* and *Double Straight*, which are relatively harder tasks for FabricFlowNet and Fabric-VSF, GPT-Fabric can obtain better

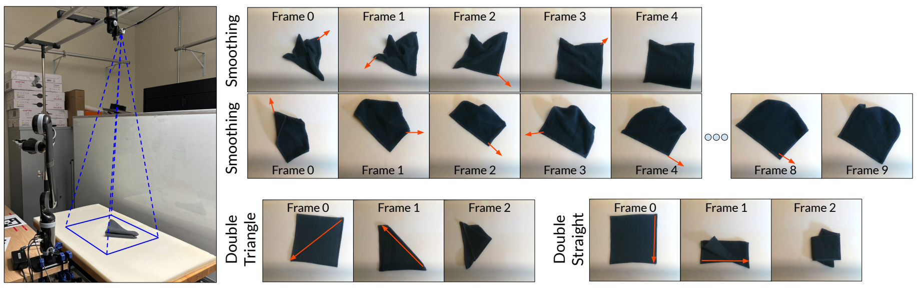


Fig. 6: **Left**: physical setup for fabric manipulation experiments. The Kinova Jaco is next to the flat workspace with blue lines indicating the top-down camera view from the Intel RealSense camera. **Right**: two smoothing rollouts and two folding rollouts of different targets (*Double Triangle* and *Double Straight*). While both folds achieve reasonable final performance, GPT-Fabric only uses two actions for the *Double Straight* fold, skipping an intermediate step shown in the subgoals (see Figure 2).

performance even in the zero-shot setting. Furthermore, performing in-context learning with either GPT-4 or GPT-3.5 generally lowers the mean particle distance errors. In particular, using just 20 demonstrations lowered the mean particle distance error to almost half for the *Double Triangle* fold.

On the other hand, Table 2 suggests that *Corners Edges Inward* and *All Corners Inward* folds are challenging for GPT-Fabric, likely due to implicit biases against the fabric center in our tests. We perform prompt ablation studies for GPT-Fabric, demonstrating how different prompt components influence folding performance, including the impact of certain biases. We defer the details to the supplementary material. Please refer to the project website for the generated folding instructions.

Overall, compared to FabricFlowNet, GPT-Fabric obtains competitive, if not better, mean particle distance error. However, GPT-Fabric only performs single-arm folding, while FabricFlowNet elegantly supports bimanual manipulation, which is not tested in our experiments or in Mo et al. [31]. We leave a "bimanual GPT-Fabric" extension to future work.

7 Physical Experiments

We perform physical experiments with GPT-Fabric (zero-shot) for smoothing and GPT-Fabric (in-context) for folding. This does *not* require additional real world data. Compared to simulation, we adjust how we detect fabric keypoints (e.g., to account for noisy depth cameras) and a camera-robot calibration procedure to convert pixels to "world positions." See the supplementary material for details. We set up a workstation with a 1-inch thick foam which has dimension 60 cm by 105 cm. We mount an Intel RealSense d435i RGBD camera at a height of about 1 m (see Figure 6). We use a 6-DOF Kinova Jaco robot manipulator with a KG-3 gripper, where we remove one of the fingers to make it similar to a standard parallel-jaw gripper as used in prior work on fabric manipulation. We use a square 30.5 cm by 30.5 cm grey dish cloth, which is similar to the fabric sizes in [31, 41].

7.1 Experiment Protocol

For smoothing and folding, a human initializes a crumpled fabric and a flat fabric, respectively. We perform 10 rollouts for smoothing and 12 rollouts for folding, three times for each of the four folding targets. Unlike prior work [31, 41], we use subgoal images from simulation instead of real-world fabric. Each action involves the robot going to a "pre-pick" pose 5 mm above the target, then lowering and grasping. Then, the robot lifts by 5 mm, moves to the "pre-place" pose, then lowers and releases its grip. The maximum number of actions per rollout is either 10 (for smoothing) or is dependent on the number of subgoals (for folding).

7.2 Physical Results

We show qualitative rollouts of our method for smoothing and folding in Figures 1 and 6. The results suggest that GPT-Fabric can achieve qualitatively acceptable fabric smoothing and folding results, despite how we do not directly train it on a fabric-related dataset. For smoothing, we obtain 0.806 ± 0.295 NI over the 10 rollouts, of which 6 also got at least 0.85 coverage. For folding, GPT-Fabric gets a 2/3 success rate for Double Triangle, and a 2/3 success rate for Double Straight. It does not succeed on the other two fold types since GPT frequently fails to set the fabric's center as the placing point, consistent with our discussion in Section 6.2. A real-world folding episode is considered successful if the system achieves the desired fabric configuration within the maximum number of actions.

We categorize failures in three cases: (1) hardware limits (including robot-camera calibration or robot imprecision), (2) corner detection errors, or (3) GPT-Fabric limitations due to GPT's incorrect reasoning. For instance, the robot can be off by 1 cm even after careful calibration or may fail to reach picking points due to range limits. In instances where fabric corners are tucked underneath, the corner detection of GPT-Fabric may fail to detect them. Finally, even with accurate corner detection, GPT may suggest suboptimal picking points (e.g., see Figure 7).

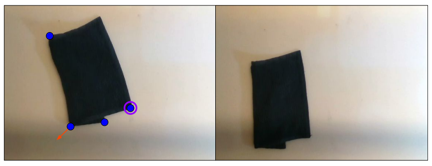


Fig. 7: An example inefficiency of GPT-Fabric from GPT. Among detected corners (blue circles), GPT-Fabric chose to pull at the bottom left corner. This barely adjusts the coverage, in contrast to potentially pulling the purple bordered point.

8 Conclusion

We present GPT-Fabric, a system for using GPT for low-level fabric manipulation. Our experiments show that GPT-Fabric can attain competitive or better results versus most prior methods, without explicitly training on a fabric-related dataset. In future work, we will extend GPT-Fabric to other deformable object manipulation tasks. We hope this work motivates research on using GPT, as well as other current and future foundation models, for low-level motor control skills for complex robotic manipulation tasks.

References

[1] D. Blanco-Mulero, O. Barbany, G. Alcan, A. Colomé, C. Torras, and V. Kyrki, "Benchmarking the Sim-to-Real Gap in Cloth Manipulation," in *IEEE Robotics and Automation Letters (RA-L)*, 2024.

[2] R. Bommasani *et al.*, "On the opportunities and risks of foundation models," *ArXiv*, 2021.

[3] J. Borràs, G. Alenyà, and C. Torras, "A Grasping-Centered Analysis for Cloth Manipulation," *IEEE Transactions on Robotics*, 2020.

[4] A. Canberk *et al.*, "Cloth Funnels: Canonicalized-Alignment for Multi-Purpose Garment Manipulation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[5] Y. Deng, K. Mo, C. Xia, and X. Wang, "Learning Language-Conditioned Deformable Object Manipulation with Graph Dynamics," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[6] A. Doumanoglou, A. Kargakos, T.-K. Kim, and S. Malassiotis, "Autonomous Active Recognition and Unfolding of Clothes Using Random Decision Forests and Probabilistic Planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[7] D. Driess *et al.*, "PaLM-E: An Embodied Multimodal Language Model," in *International Conference on Machine Learning (ICML)*, 2023.

[8] Z. Erickson, H. Clever, G. Turk, C. K. Liu, and C. Kemp, "Deep Haptic Model Predictive Control for Robot-Assisted Dressing," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[9] Z. Erickson, V. Gangaram, A. Kapusta, C. K. Liu, and C. C. Kemp, "Assistive Gym: A Physics Simulation Framework for Assistive Robotics," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[10] R. Firoozi *et al.*, "Foundation Models in Robotics: Applications, Challenges, and the Future," *arXiv preprint arXiv:2312.07843*, 2023.

[11] A. Ganapathi *et al.*, "Learning Dense Visual Correspondences in Simulation to Smooth and Fold Real Fabrics," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[12] G. T. Google, "Gemini: A Family of Highly Capable Multimodal Models," *arXiv preprint arXiv:2312.11805*, 2023.

[13] C. Harris and M. Stephens, "A Combined Corner and Edge Detector," in *In Proceedings of the Fourth Alvey Vision Conference*, 1988.

[14] R. Hoque *et al.*, "VisuoSpatial Foresight for Multi-Step, Multi-Task Fabric Manipulation," in *Robotics: Science and Systems (RSS)*, 2020.

[15] R. Hoque *et al.*, "Learning to Fold Real Garments with One Arm: A Case Study in Cloud-Based Robotics Research," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[16] Y. Hu *et al.*, "Toward General-Purpose Robots via Foundation Models: A Survey and Meta-Analysis," *arXiv preprint arXiv:2312.08782*, 2023.

[17] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents," in *International Conference on Machine Learning (ICML)*, 2022.

[18] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, "VoxPoser: Composable 3D Value Maps for Robotic Manipulation with Language Models," in *Conference on Robot Learning (CoRL)*, 2023.

[19] K. Kawaharazuka, T. Matsushima, A. Gambardella, J. Guo, C. Paxton, and A. Zeng, "Real-World Robot Applications of Foundation Models: A Review," *arXiv preprint arXiv:2402.05741*, 2024.

[20] T. Kwon, N. D. Palo, and E. Johns, "Language Models as Zero-Shot Trajectory Generators," in *IEEE Robotics and Automation Letters (RA-L)*, 2024.

[21] R. Lee, D. Ward, A. Cosgun, V. Dasagi, P. Corke, and J. Leitner, "Learning Arbitrary-Goal Fabric Folding with One Hour of Real Robot Experience," in *Conference on Robot Learning (CoRL)*, 2020.

[22] J. Liang *et al.*, "Code as Policies: Language Model Programs for Embodied Control," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[23] X. Lin, Y. Wang, Z. Huang, and D. Held, "Learning Visible Connectivity Dynamics for Cloth Smoothing," in *Conference on Robot Learning (CoRL)*, 2021.

[24] X. Lin, Y. Wang, J. Olkin, and D. Held, "SoftGym: Benchmarking Deep Reinforcement Learning for Deformable Object Manipulation," in *Conference on Robot Learning (CoRL)*, 2020.

[25] M. Lippi *et al.*, "Latent Space Roadmap for Visual Action Planning of Deformable and Rigid Object Manipulation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[26] Y. J. Ma *et al.*, "Eureka: Human-Level Reward Design via Coding Large Language Models," in *International Conference on Learning Representations (ICLR)*, 2024.

[27] J. Maitin-Shepard, M. Cusumano-Towner, J. Lei, and P. Abbeel, "Cloth Grasp Point Detection Based on Multiple-View Geometric Cues with Application to Robotic Towel Folding," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[28] J. Mao, Y. Qian, J. Ye, H. Zhao, and Y. Wang, "GPT-Driver: Learning to Drive with GPT," *arXiv preprint arXiv:2310.01415*, 2023.

[29] J. Mao, J. Ye, Y. Qian, M. Pavone, and Y. Wang, "A Language Agent for Autonomous Driving," *arXiv preprint arXiv:2311.10813*, 2023.

[30] S. Miller, J. van den Berg, M. Fritz, T. Darrell, K. Goldberg, and P. Abbeel, "A Geometric Approach to Robotic Laundry Folding," in *International Journal of Robotics Research (IJRR)*, 2012.

[31] K. Mo, C. Xia, X. Wang, Y. Deng, X. Gao, and B. Liang, "Foldsformer: Learning Sequential Multi-Step Cloth Manipulation With Space-Time Attention," in *IEEE Robotics and Automation Letters (RA-L)*, 2023.

[32] OpenAI, “GPT-4 Technical Report,” *arXiv preprint arXiv:2303.08774*, 2023.

[33] G. Salhotra, I.-C. A. Liu, M. Dominguez-Kuhne, and G. S. Sukhatme, “Learning Deformable Object Manipulation from Expert Demonstrations,” in *IEEE Robotics and Automation Letters (RA-L)*, 2022.

[34] J. Sanchez, J.-A. Corrales, B.-C. Bouzgarrou, and Y. Mezouar, “Robotic Manipulation and Sensing of Deformable Objects in Domestic and Industrial Applications: a Survey,” in *International Journal of Robotics Research (IJRR)*, 2018.

[35] D. Seita *et al.*, “Deep Imitation Learning of Sequential Fabric Smoothing From an Algorithmic Supervisor,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[36] J. Shi and C. Tomasi, “Good Features to Track,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1994.

[37] I. Singh *et al.*, “ProgPrompt: Program generation for situated robot task planning using large language models,” *Autonomous Robots (AURO)*, 2023.

[38] E. Torgerson and F. Paul, “Vision Guided Robotic Fabric Manipulation for Apparel Manufacturing,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 1987.

[39] Y.-J. Wang, B. Zhang, J. Chen, and K. Sreenath, “Prompt a Robot to Walk with Large Language Models,” *arXiv preprint arXiv:2309.09969*, 2023.

[40] Y. Wang *et al.*, “RL-VLM-F: Reinforcement Learning from Vision Language Foundation Model Feedback,” in *International Conference on Machine Learning (ICML)*, 2024.

[41] T. Weng, S. Bajracharya, Y. Wang, K. Agrawal, and D. Held, “FabricFlowNet: Bimanual Cloth Manipulation with a Flow-based Policy,” in *Conference on Robot Learning (CoRL)*, 2021.

[42] B. Willimon, S. Birchfield, and I. Walker, “Model for Unfolding Laundry using Interactive Perception,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[43] Y. Wu, W. Yan, T. Kurutach, L. Pinto, and P. Abbeel, “Learning to Manipulate Deformable Objects without Demonstrations,” in *Robotics: Science and Systems (RSS)*, 2020.

[44] Z. Xu, C. Chi, B. Burchfiel, E. Cousineau, S. Feng, and S. Song, “DextAIRity: Deformable Manipulation Can be a Breeze,” in *Robotics: Science and Systems (RSS)*, 2022.

[45] W. Yan, A. Vangipuram, P. Abbeel, and L. Pinto, “Learning Predictive Representations for Deformable Objects Using Contrastive Estimation,” in *Conference on Robot Learning (CoRL)*, 2020.

[46] J. Yang, H. Zhang, F. Li, X. Zou, C. Li, and J. Gao, “Set-of-Mark Prompting Unleashes Extraordinary Visual Grounding in GPT-4V,” *arXiv preprint arXiv:2310.11441*, 2023.

[47] W. Yu *et al.*, “Language to Rewards for Robotic Skill Synthesis,” in *Conference on Robot Learning (CoRL)*, 2023.

Supplementary Material for GPT-Fabric: Smoothing and Folding Fabric by Leveraging Pre-Trained Foundation Models

A Additional Prompt Details

This section includes a description of the reasoning and intuition behind the development of the prompts and the overall structure of the GPT-Fabric pipeline. The full prompts can be found on the project website[1].

A.1 GPT-Fabric for Smoothing

The design of our high-level smoothing strategy incorporates human intuition about smoothing. We believe that repeatedly choosing the same corner to pick can drag the fabric out of the camera's view, as well as picking and placing opposite corners back-to-back, which simply moves the fabric back and forth. When the picking point is chosen, an efficient direction to pull it can be the vector starting from the center point of the fabric and pointing towards the chosen picking point.

We also employ a verification procedure, visualized in Figure 4 of the main text, to align the output of GPT-4V with the high-level strategy. We first perform a *proximity check* to ensure that the chosen picking point is sufficiently far from the last step's picking point (or its symmetric point). If this passes, we then conduct a *direction check* to verify whether the predicted moving direction significantly deviates from the actual direction which originates from the center point and extends to the picking point. We take the prediction of GPT-4V as the final output if both tests pass. Otherwise, we provide GPT-4V with a correction message explaining why the previous prediction fails, and the visualization of the output action to the original input and let it infer again. If GPT-4V's output fails to pass for three consecutive times, we discard the picking point's proximity check and only perform the direction check.

If the detected corners are all near the prior step's picking point and its symmetric point, this can make it impossible to pass the proximity check. However, the direction check is essential, as a "wrong" direction can be catastrophic. In our cases, GPT-4V occasionally makes the mistake of choosing the exact opposite direction, pulling the chosen picking point towards the center point. Therefore, we still conduct the direction check, believing that pulling the chosen picking point away from the center point will not worsen the situation as it will likely stretch the fabric to the pulling direction.

[1] https://tinyurl.com/gptfab

A.2 GPT-Fabric for Folding

Being consistent with prior work, we assume access to just the folding subgoal images, without any textual details corresponding to the folding task. Since GPT-Fabric is dependent on the vision reasoning abilities of GPT-4V for generating the folding instructions, we require that the generated instructions are reasonably accurate for the downstream LLM to obtain the pick and place actions effectively.

In preliminary experiments, we observed that the model exhibited errors ranging from disregarding all the information in the subgoal images by interpreting them as placeholder thumbnails to being biased towards suggesting diagonal folds or center folds in the generated folding instructions. To address these issues, we construct a language prompt with five major components:

1. *Visual Context*: A brief explanation of the context represented by the subgoal images to ensure that GPT-4V treats them as valid image input.
2. *Pick-and-place Arrow*: A description of the pick-and-place action facilitated by a black arrow attached to the subgoal images depicting a folding step.
3. *Center Pivoting*: A reasoning logic that utilizes the action description specified above to estimate the possible place point location relative to the fabric center to ensure that the place point is not biased towards being one of the fabric corners.
4. *Instruction Constraint*: An explicit requirement to generate the folding instruction in terms of the pick and place point to ensure that the downstream LLM could utilize that to return an executable action.
5. *Output Format*: An explicit output format specification that could be parsed to get an instruction to be used by the downstream LLM.

See Section B.2 for our experiments on ablating these five prompt components.

For the downstream LLM (either GPT-4 or GPT-3.5) to effectively utilize this folding instruction, we construct an additional natural language prompt to describe the task, ensuring that the model is aware of the purpose of the context while generating an output that follows a fixed format in order to parse an executable action. We use a chain-of-thought prompting mechanism to have the LLM produce the reasoning behind the generated pick-and-place action.

B Additional Experiment Details

B.1 Simulation Experiments: Ablations for Smoothing

The main text discusses our ablations with GPT-Fabric for smoothing, along with the experimental results. The ablations are:

– *Random discretized action:* We conduct the corner detection to the raw image observation $\mathbf{o}_t$ and randomly choose one from those detected corners as the picking point x_{pick}. We also randomly choose the pulling action's direction and distance from the same discretized list in the prompt (see Section 4.2 in the main text). This is a way to compare how much GPT outperforms random guessing.

- *No image preprocessing & no eval. module:* We only conduct the corner detection to the raw image observation $\mathbf{o}_t$ and provide the detected corners' coordinates to GPT; we adapt the system and user prompt to the removal of the image preprocessing module. We also remove the evaluation module and directly execute the action from GPT.
- *No eval. module:* We remove the evaluation module.
- *No verifying pick point proximity in eval. module:* In the evaluation module, we only perform the *direction check* to verify GPT's output action's direction and execute the action if it passes.

Our results in the main text show that removing any major component in GPT-Fabric for leads to significantly worse smoothing performance. The performance of *No image preprocessing & no eval. module* is worse than *Random discretized action*, suggesting that fabric smoothing is an extremely challenging task for GPT alone, underlining the importance of our image preprocessing and evaluation modules. We also witness worse performance in *No eval. module* where GPT occasionally pulls the picking point towards the center point, resulting in an enormous coverage decrease.

Method	Double Triangle	All Corners Inward
No Visual Context	58.88 ± 21.44	95.75 ± 22.07
No Pick-and-place Arrow	97.53 ± 11.49	72.79 ± 20.06
No Center Pivoting	65.29 ± 19.05	148.57 ± 28.60
No Instruction Constraint	58.88 ± 24.92	68.42 ± 16.76
No Output Format	87.85 ± 18.15	36.09 ± 14.55
GPT-Fabric (zero-shot)	68.52 ± 20.42	75.59 ± 23.11

Table 4: Mean particle distance errors (mm) for GPT-Fabric prompt ablations for folding in simulation. Lower numbers are better. The last row refers to our full method.

B.2 Simulation Experiments: Ablations for Folding

We ablate five core components of the prompt (see Section A.2). We perform these ablations for *Double Triangle* and *All Corners Inward* folding using a similar experimental setup as discussed in the main text. For these experiments, we average over three trials of GPT-Fabric (zero-shot) and use GPT-4 as the downstream LLM to generate executable pick and place points. The results are in Table 4.

Our results show that removing any major component leads to a performance degradation in at least one folding type in general, highlighting the importance of all five components. This benefit could be due to biases in GPT-Fabric predictions in the absence of certain components. For instance, on ablating the *Output Format* component, we observe that GPT-Fabric often chooses the fabric center as the placing point for the folding actions, leading to a significantly lower mean particle distance error for *All Corners Inward* fold but a higher mean particle distance error for *Double Triangle* fold when compared with GPT-Fabric (zero-shot). The results for GPT-Fabric (zero-shot) in Table 4 differ from the results for GPT-Fabric (GPT-4, zero-shot) in the Table 2 of the main text, due to different number of trials and since GPT is nondeterministic even with temperature of 0.

B.3 Simulation Experiments: Additional Results

We report the normalized coverage in Table 5. This is an alternative metric (also used in [23]) compared to normalized improvement, which we report in the main text. These metrics under consideration are evaluated on the same set of experiments.

Method	# of actions: 05	# of actions: 10	# of actions: 20	Data Size
VCD†	0.776 ± 0.132	0.872 ± 0.128	0.985 ± 0.187	2K
VCD§	0.643 ± 0.121	0.745 ± 0.101	0.876 ± 0.151	2K
VCD, graph imitation†	0.837 ± 0.150	0.966 ± 0.236	0.996 ± 0.076	2K
GPT-Fabric (zero-shot)	0.887 ± 0.120	0.980 ± 0.112	0.992 ± 0.036	0

†Reported by VCD authors. §Using trained model weights provided by VCD authors.

Table 5: Extended results for fabric smoothing in simulation, comparing GPT-Fabric versus prior works on the same starting crumpled (square) fabric configurations from Lin et al. [23]. We format the table in a similar manner as in the main text, except that we report *normalized coverage* instead of *normalized improvement*, and we only compare GPT-Fabric with VCD variants.

B.4 Real World Experiment Setup

As mentioned in the main text, we incorporate some changes in our system, specifically for preprocessing the input image, when compared with simulation. Limited by the accuracy of the depth camera in the real world, we do not utilize depth information for assistance and instead use the top-down RGB image directly. We use a contour-finding algorithm to obtain the contour of the fabric and approximate the achieved coverage s by its area. We construct a bounding box around the contour and use its center point to approximate the center point of the fabric. We apply Harris Corner Detector [13] for folding tasks and Shi-Tomasi Corner Detector [36] for smoothing tasks to the top-down RGB image, retaining only the detected corners within the bounding box as candidate picking points. We use a standard calibration procedure to convert the "camera position" to the "world position" for the robot in both tasks, converting the picking and placing points computed by GPT-Fabric to real world positions, where the robot can execute its actions.

B.5 Inference Time

Using our setup of two NVIDIA RTX 4090 GPUs, the average inference time for planning one pick-and-place smoothing action with GPT-Fabric in simulation is approximately 18.01 s, which is slower compared to 12.70 s for VCD (as reported by the authors using their setup of four NVIDIA RTX 2080Ti GPUs). Similarly, the average inference time for planning one pick-and-place folding action with GPT-Fabric is approximately 12.46 s, which is slower compared to 0.02 s for Foldsformer [31] when replicated on our setup. These numbers are partially due to querying the OpenAI API multiple times in both workflows. In future work we will explore reducing the number of API queries.

B.6 Budget

Running experiments in this paper involved frequent calls to the OpenAI API to use GPT models. In all, we paid approximately 2300 USD for the API, which includes the preliminary trials and all the reported experiments in simulation and in real.